DEGREE PROJECT, IN COMPUTER SCIENCE , FIRST LEVEL
STOCKHOLM, *SWEDEN 2015*

Automatic Scheduling for schools

SCALABILITY OF THE HOPFIELD NEURAL NETWORKS

DIEGO MARTÍNEZ MARRODÁN

KTH ROYAL INSTITUTE OF TECHNOLOGY

CSC SCHOOL

KTH Computer Science
and Communication

Automatic Scheduling for schools

Scalability of the Hopfield Neural Networks

DIEGO MARTÍNEZ MARRODÁN

Degree Project in Computer Science, DD143X
Supervisor: Pawel Herman
Examiner: Örjan Ekeberg

CSC, KTH 2015-04

Abstract

This project is focused on evaluating, in terms of real time needed to find a solution, the scalability of Hopfield Neural Networks, a Machine Learning method, applied to a common problem that every educational institution has to deal with at least once in every academic year, timetabling.

With this purpose, the problem is first introduced. And secondly, in the background, the concept of "constraint" is presented, to continue with a brief explanation of Artificial Neural Networks, the state of the art and more specifically, how Hopfield Neural Networks are characterized.

The formulation, modifications used, and the algorithm are presented. This algorithm will be implemented in MATLAB, and it will be run on data sets of different sizes.

The results obtained for the presented data sets are presented in a table and graphs, to later discuss these results. In this discussion, it is found that the time spent to get a solution could scale quadratically with respect to the size of the problem, but there is not statistical evidence to this hypothesis.

Finally, the conclusion is that Hopfield Neural Networks could have a good scalability if the hypothesis worked for bigger data sets, and some future work in the field is presented, like using sparse matrices for the implementation of the problem, or studying the scalability of Hopfield Neural Networks in other kinds of scheduling.

Contents

Chapter 1

Introduction

One of the problems that we have to solve every day with a higher or lower success is organizing our schedule or scheduling. We have a set of activities that we have or would like to do, and a limited amount of resources (space, time, people, capital, etc.) for doing them. So we try to organize them in order to use the minimum amount of resources required.

This problem can be applied to different kinds of scheduling, like job shop scheduling, transportation scheduling, broadcast scheduling, academic scheduling, etc. This project will be focused on the academic scheduling, specifically applied to schools.

Schools have a limited amount of teachers, and classrooms, for a number of registered students that are assigned to different classes. Schedules have to be organized in such a way that students can attend all the lectures of the courses that they have to take (which implies that there should be no conflicts between the lectures of the courses that a class has to attend), having in mind constraints like the limited number of classrooms, and that a teacher cannot deliver two lectures at the same time.

Every resource has an associated cost, in time and capital. So there is a demand for automated scheduling that can perform this task by optimising the resources, in order to minimize the total cost. For this reason, there have been many investigations on the field with different approaches [1, 3, 5, 6, 8]. This project will be focused on a Machine Learning approach to the problem, using Hopfield Neural Networks.

Every year, schools have to design a schedule for the taught courses that matches students with teachers into the available classrooms avoiding as many collisions as possible. Traditionally, this task has been done manually and it can take up to several weeks in the case of university scheduling [7].

Scheduling is a complex problem, usually a NP-Complete problem [2, 4], and therefore traditional approaches to solve it (simulation models, analytical models, heuristic approaches...) are quite inefficient [1, 2].

Solving the problem by applying Machine Learning techniques has been proven

to be quite effective [2], and therefore some algorithms for this have been developed and evaluated. Nevertheless, there are not many evaluations of them in terms of their efficiency when they are applied to data sets of different sizes, or the variation of the size is proportional in all of the dimensions of the data sets [6].

1.1 Aims

This project will be focused on testing the behaviour of a Hopfield Neural Network algorithm [6] on school requirement data sets of different size (in terms of groups of students, teachers, and classrooms available, having in mind the total and the relative amount of these resources, which are the dimensions of the problem), in order to find out how its performance scales with the size of the problem.

The efficiency of the Hopfield Neural Network algorithm will be defined in terms of computational speed in time (real time needed to find a solution in a given computer since the algorithm starts to run), considering that the results should have enough quality. This means that the number of constraints violated in the generated schedules should be minimum (what is meant by constraints, is defined in Chapter 2).

Thus, what the project will try to find out is how the time needed for finding a solution to the problem increases or decreases when the amount of each resource (dimensions of the problem) of it is bigger or smaller, respectively. And also whether increasing or decreasing a specific resource (groups of students, teachers or classrooms available in this case), has more or less influence on this time than increasing or decreasing a different resource.

1.2 Scope

In the project, school scheduling is chosen over university scheduling, among other reasons, because in schools, students are assigned to classes that usually take the same courses; while in universities, students have more freedom to select courses, and all (or most) of them should be considered when designing the timetable, increasing a lot the size of the problem. Choosing school scheduling, the problem is simplified and its size is smaller.

Therefore, for this reason, students will be treated as groups and not as individuals, assuming that all students in a group have the same lectures at the same time and place and move together as a group.

In addition, the schedule generated will be a weekly schedule that will be repeated during the term or academic year. This weekly schedule will consist of 30 periods, which can be easily distributed in 5 days in which each day has 6 periods.

This problem has four dimensions: classes or groups of students, teachers, classrooms or venues, and periods, being this last one fixed. Consequently, the set C of groups of students, the set T of teachers, the set V of venues, and the set P of regular periods or time-slots (of dimensions c, t, v and p respectively) are known. Also, it

will be assumed that there is available a matrix of requirements R of dimensions $c \times t \times v$ in which R_{ijk} indicates how many times the group i has to meet the teacher j in the classroom k. Thus, the functionality of the algorithm will be to distribute those requirements over the set P.

It is not a goal of this project to obtain a perfect solution (a solution without conflicts), and either to apply the algorithm to real data sets. The algorithm will be executed on three fictional data sets obtained from the OR-Library at Imperial College London (`http://www.ms.ic.ac.uk`), and seven more modified from these ones in order to have diversity in the relative and total amount of the resources available.

1.3 Outline

After this introduction, following with the background, Artificial Neural Networks are defined and the state of the art is presented. Then, in the methods chapter, the formulation, algorithm, data sets, and settings used are shown.

Next in the report, the results obtained from the execution of the algorithm in the different data sets, are displayed in a table and graphics. Continuing with a discussion of these results, looking for an answer to the questions raised in the introduction. And finally, the report ends with the conclusions obtained.

Chapter 2

Background

2.1 Constraints in timetabling

When designing a timetable for schools, lectures have to be scheduled following some limitations called "constraints". Usually, two kinds of constraints are defined for this problem:

- **Hard constraints**: Constraints that cannot be violated in any way. A solution that does not violate any hard constraint is called a valid or feasible solution.
- **Soft constraints**: Constraints that are desirable, but not mandatory. The less soft constraints violated, the better the solution will be.

There is not any global agreement about hard and soft constraints for school scheduling, and different institutions usually have different requirements [7]. Therefore, there are different hard and soft constraints considered among the literature [5, 6, 8, 9, 10], but usually there are more similarities on the hard constraints. Some examples of hard and soft constraints are the following:

Hard constraints

- No students or teachers attend more than one event at the same time.
- There is only one event in each room at any time slot.
- The room is big enough for all the attending students.

Soft constraints

- A student or teacher has not more than an established number of lessons consecutively.
- A student or teacher has not more than an established number of lessons on the same day.
- A student or teacher has not big gaps between lessons.

For the project, only the two first hard constraints will be considered, since its goal, as stated in chapter 1, is to look into the efficiency of the applied algorithm, and not generating perfect or realistic schedules. Therefore, this can be done avoiding a complex implementation.

2.2 Artificial Neural Networks

Artificial Neural Networks are a family of statistical learning algorithms inspired by biological neural networks. In them, there are nodes called "neurons" that are connected together to form a network. They are essentially mathematical models defining a function $f : X \to Y$, or a distribution over X or both X and Y. And are typically defined by:

- The interconnection layer pattern between the different layers of neurons.
- The learning process for updating the weights of the interconnections.
- The activation function that converts a neuron's weighted input to its output activation.

In the field of school timetabling, there have been some investigations and comparisons between Neural Networks and other methods to solve the problem [6, 9, 10]. The implementation used in this project will be based on the algorithm presented in [6] (and in chapter 3), using a Hopfield Neural Network.

2.2.1 Hopfield Neural Networks

Hopfield Neural Networks are characterized by being a fully interconnected network with N neurons. Neuron i has a net input value or internal state u_i and an output value v_i (which can have a real value bounded between 0 and 1 in the continuous model, or a binary value in the discrete model, being this last one the one which is going to be used in this project). Fig. (1) represents a Hopfield Neural Network consisting of four neurons. The internal state u_i is obtained from the sum of the multiplication of the output value of all the other neurons v_j, by the strength of the connections from neuron i to j, given by W_{ij} (where $W_{ii} = 0$, and $W_{ij} = W_{ji}$), and then adding a bias current (or additional input) I_i [6, 9].

$$u_i = \sum_{j=1}^{N} W_{ij} v_j + I_i \tag{1}$$

The relationship between the internal state u_i and its output value v_i is given by an activation function $g(u_i)$. Which is the following in the discrete model:

$$v_i = g(u_i) = \begin{cases} 1, & \text{if } u_i > 0 \\ 0, & \text{if } u_i \leq 0 \end{cases} \tag{2}$$

The neurons are updated in following steps (sequentially or in parallel) following the next rules [6]:

$$u_i(t+1) = u_i(t) + \Delta t \left(\sum_{j=1}^{N} W_{ij} v_j + I_i \right) \tag{3}$$

$$v_i(t+1) = g(u_i) \tag{4}$$

Where Δt is a constant time-step, that will be 1 in this project. Applying this updates, the network will converge to a local minimum of the following energy function:

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} W_{ij} v_i v_j - \sum_{i=1}^{N} I_i v_i \tag{5}$$

This energy is a real value that increases with the number of constraints violated, so the higher this value is, the more constraints that are being violated, which results in a worse schedule. Therefore, a perfect schedule would have an energy value of 0. Nonetheless, as stated in section 1.2, it is not a goal of the project to obtain schedules with energy 0.

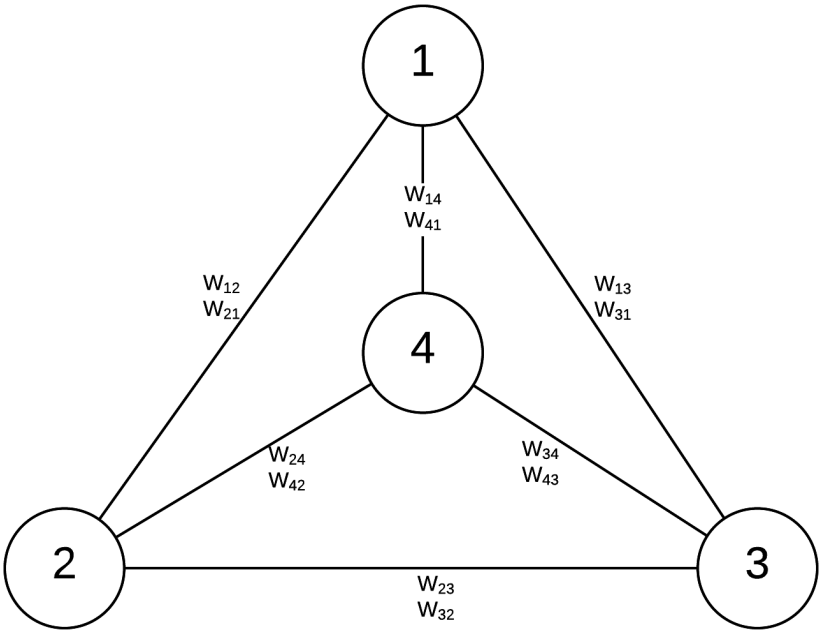

Figure 1. Example of Hopfield Neural Network with four nodes

Chapter 3

Methods

The implementation of the Hopfield Neural Network algorithm is going to be for MATLAB, since there is a lot of work with matrices and MATLAB offers many tools to manage them, making the task easier and more efficient.

The data sets that are going to be used are based on the ones publicly available from the OR-Library at Imperial College London (`http://www.ms.ic.ac.uk`). Concretely, the original data sets hdtt4 (4 groups, 4 teachers and 4 venues), hdtt6 (6 groups, 6 teachers and 6 venues), and hdtt8 (8 groups, 8 teachers and 8 venues), are going to be used and derived into another 7 different datasets. This way, the number of groups, teachers and venues does not increase or decrease in the same amount and it will be possible to show if any variable is more relevant than the others when scaling the problem, and also, use the algorithm with a bigger problem than the original data set hddt8 (hddt10). Since the number of periods considered is going to be 30 for every week, the number of neurons, with the datasets used, will range from 1 920 neurons for the hdtt4 set to 30 000 for the hdtt10 set. The base data sets are totally constrained, which means that each period contains every class, teacher and venue. This makes it extremely difficult to find a perfect solution (solution with energy 0).

3.1 Formulation

The resulting schedule will be represented as a boolean matrix of dimensions $c \times t \times v \times p$ defined as follows:

$$x_{ijkl} = \begin{cases} 1, & \text{if group } i \text{ and teacher } j \text{ are assigned to venue } k \text{ in period } l \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The formulation used is the following:

$$\sum_{l=1}^{p} x_{ijkl} = R_{ijkl} \quad (\forall i \in C; j \in T; k \in V) \tag{7}$$

$$\sum_{j=1}^{t}\sum_{k=1}^{v} x_{ijkl} \leq 1 \quad (\forall i \in C; l \in P) \tag{8}$$

$$\sum_{i=1}^{c}\sum_{k=1}^{v} x_{ijkl} \leq 1 \quad (\forall j \in T; l \in P) \tag{9}$$

$$\sum_{i=1}^{c}\sum_{j=1}^{t} x_{ijkl} \leq 1 \quad (\forall k \in V; l \in P) \tag{10}$$

$$x_{ijkl} = 0 \; or \; 1 \quad (\forall i \in C; j \in T; k \in V; l \in P) \tag{11}$$

The Equation (7) ensures that the requirements (contained in the matrix R) are satisfied. Equations (8), (9) and (10) prevent class, teacher, and venue clashes respectively. And finally, Equation (11) is needed to ensure the integrity of the solutions.

3.2 Hopfield Neural Networks applied to timetabling

In this project, the timetabling problem is represented as an optimization (or minimization of the energy) problem that is going to be mapped into a Hopfield Neural Network. In order to do so, it is needed to express the constraints of the problem as penalty terms into the energy function that has to be minimized [6, 9]. This energy function, can be expressed as in [6]:

$$\begin{aligned} E = {} & \frac{\alpha}{2}\sum_{i=1}^{c}\sum_{j=1}^{t}\sum_{k=1}^{v}\left(\sum_{l=1}^{p} x_{ijkl} - R_{ijk}\right)^2 + \frac{\beta}{2}\sum_{i=1}^{c}\sum_{j=1}^{t}\sum_{k=1}^{v}\sum_{l=1}^{p}\sum_{\substack{j'=1\\ j'\neq j}}^{t}\sum_{\substack{k'=1\\ k'\neq k}}^{v} x_{ijkl}x_{ij'k'l} \\ & + \frac{\chi}{2}\sum_{i=1}^{c}\sum_{j=1}^{t}\sum_{k=1}^{v}\sum_{l=1}^{p}\sum_{\substack{i'=1\\ i'\neq i}}^{c}\sum_{\substack{k'=1\\ k'\neq k}}^{v} x_{ijkl}x_{i'jk'l} + \frac{\gamma}{2}\sum_{i=1}^{c}\sum_{j=1}^{t}\sum_{k=1}^{v}\sum_{l=1}^{p}\sum_{\substack{i'=1\\ i'\neq i}}^{c}\sum_{\substack{j'=1\\ j'\neq j}}^{t} x_{ijkl}x_{i'j'kl} \end{aligned} \tag{12}$$

Where α, β, χ and γ are the penalty parameters which have to be tuned to equally balance the terms of the function. The first term in the Equation (12) is zero when Equation (7) is fulfilled. The other three terms, will be zero when the Equations (8), (9), and (10) are satisfied, and otherwise, it will represent one or more student, teacher or venue clashes respectively.

With the right network weights and external bias, a Hopfield network can be designed to minimize this function. Considering the matrix of neurons v equivalent

to the solution matrix x, we have a eight-dimensional matrix of weights W since it connects all neurons v_{ijkl} to all other neurons $v_{i'j'k'l'}$. Now, applying Equation (5) to this problem, the energy function would be:

$$E = -\frac{1}{2}\sum_{i=1}^{c}\sum_{j=1}^{t}\sum_{k=1}^{v}\sum_{l=1}^{p}\sum_{i'=1}^{c}\sum_{j'=1}^{t}\sum_{k'=1}^{v}\sum_{l'=1}^{p} W_{ijkl,i'j'k'l'} v_{ijkl} v_{i'j'k'l'} - \sum_{i=1}^{c}\sum_{j=1}^{t}\sum_{k=1}^{v}\sum_{l=1}^{p} I_{ijkl} v_{ijkl} \tag{13}$$

At this point, the matrix W and the matrix of external bias I, can be obtained expanding and rearranging the function (12), and comparing it with the function (13), as it is explained in [6]:

$$\begin{aligned} W_{ijkl,i'j'k'l'} = &- \alpha\delta_{ii'}\delta_{jj'}\delta_{kk'} \; - \; \beta\delta_{ii'}(1-\delta_{jj'})(1-\delta_{kk'})\delta_{ll'} \\ &- \; \chi(1-\delta_{ii'})\delta_{jj'}(1-\delta_{kk'})\delta_{ll'} \; - \; \gamma(1-\delta_{ii'})(1-\delta_{jj'})\delta_{kk'}\delta_{ll'} \end{aligned} \tag{14}$$

$$I_{ijkl} = \alpha R_{ijk} \tag{15}$$

Where $\delta_{yy'}$ is the Kronecker-Delta function, $\delta_{yy'} = 1$ if $y = y'$ and 0 otherwise.

There is also an additional constant term $\frac{\alpha}{2}\sum_{i=1}^{c}\sum_{j=1}^{t}\sum_{k=1}^{v} R_{ijk}^2$ to complete the equivalence, but it can be ignored since it does not influence the location of the local and global minima, and there is not a constant term in the Hopfield energy function (13).

Finally, the rule for updating the internal states and output values of the neurons, applied to this specific problem, would be the following:

$$u_{ijkl}(t+1) = u_{ijkl}(t) + \Delta t\left(\sum_{i'=1}^{c}\sum_{j'=1}^{t}\sum_{k'=1}^{v}\sum_{l'=1}^{p} W_{ijkl,i'j'k'l'} v_{i'j'k'l'} + I_{ijkl}\right) \tag{16}$$

$$v_{ijkl}(t+1) = g(u_{ijkl}) \tag{17}$$

3.2.1 Modifications and algorithm

According to [6], one of the problems of Hopfield Neural Networks is that it usually converges to local minimas while performing the descent, so it does not reach the global minima (which would be the best solution), and therefore the resulting timetable is not very good. Thus, they propose some modifications to avoid this as much as possible.

One of this modifications is terminating the updating after a small number of iterations, and then modify the output value of a neuron vaccording to a threshold between 0 and 1. So if the threshold is 0.8, there is a 20% chance that the output value will be assigned to 1, and a 80% chance that it will be assigned to 0.

The other modification proposed is to allow a particular descent to terminate prematurely if a local minima has been already found. With this purpose, a measure called *stability*, that counts the number of neurons whose output value have changed, will be used. If this number is below a predefined value, the current descent is finished, the previous modification acts and then it continues with another descent. Allowing this way to dynamically reduce the number of iterations if the stability point has been reached.

With these two modifications, the chances that the algorithm reaches a global minima and does not get trapped into a local minima are much higher. This algorithm, described also in [6], is the following:

Algorithm 1 Hopfield Neural Network algorithm

```
initialize u (randomly around zero)
calculate v using Equation (2)
for i = 1 to descents do
    for j = 1 to iterations do
        for k = 1 to N do
            update u_k using Equation (3)
            update v_k using Equations (2)
        end for
        calculate stability = sum_{k=1}^{N} Δv_k
        if stability criteria is satisfied then
            exit j loop
        end if
    end for
    randomly choose a neuron, and flip it according to threshold
    renormalize u (u_k → 1 if u_k > 0;  u_k → 0 if u_k < 0)
end for
```

In this project, the inner loop has been slightly modified in order to make it easier to map from the result into a real schedule. Therefore, instead of a loop of size N, there are four nested loops of size c, t, v and p, which does not make much difference in terms of efficiency since $N = c \cdot t \cdot v \cdot p$. And the updates for u and v will follow Equations (16) and (17) respectively.

3.3 Parameters, data sets, and hardware and software

The parameters that are going to be used need to be tuned to get proper results. After some tests over the data sets, these parameters that manage to get a low energy without skipping many requirements (which would result in a low energy value for terms 2, 3 and 4 in the Equation (12) but a high energy value for term 1), are the following:

$\alpha \to 0.2$

$\beta \rightarrow 0.1$

$\chi \rightarrow 0.1$

$\gamma \rightarrow 0.1$

$descents \rightarrow 4$

$iterations \rightarrow 10$

$stability \rightarrow 4 \quad \left(\sum_{k=1}^{N} \Delta v_k\right)$

$threshold \rightarrow 0.5$ (probability of modifying a random i neuron output value v_i to 0, otherwise to 1).

For the evaluation, the algorithm will be run in each data set 10 times, and the time that the algorithm takes in each data set is going to be considered the average of the running time of those 10 executions. After this, a relationship that correlates the running time and the size of the problem, will be tried to be found. The abbreviations that are going to be used on the graphics for the data sets evaluated are the following:

$D4 \rightarrow$ (4 groups, 4 teachers and 4 venues)

$D6 \rightarrow$ (6 groups, 6 teachers and 6 venues)

$D6\_8C \rightarrow$ (8 groups, 6 teachers and 6 venues)

$D6\_8T \rightarrow$ (6 groups, 8 teachers and 6 venues)

$D6\_8V \rightarrow$ (6 groups, 6 teachers and 8 venues)

$D8 \rightarrow$ (8 groups, 8 teachers and 8 venues)

$D8\_6C \rightarrow$ (6 groups, 8 teachers and 8 venues)

$D8\_6T \rightarrow$ (8 groups, 6 teachers and 8 venues)

$D8\_6V \rightarrow$ (8 groups, 8 teachers and 6 venues)

$D10 \rightarrow$ (10 groups, 10 teachers and 10 venues)

The hardware and software used for running this algorithm on the different data sets is a laptop with an Intel® Core™ i7-2670QM with 4 cores at 2.20GHz, and 8GB of RAM on a Windows 8.1 (64 bits) OS.

Chapter 4

Results

Table 1 shows the mean time (over 10 runs) that the algorithm took to find a solution for the different data sets, in seconds. The parameters used in the program are commented in Chapter 3.

D4	**D6**	**D6_8C**	**D6_8T**	**D6_8V**
4.3340	39.2513	68.1623	65.3964	66.1153

D8_6C	**D8_6T**	**D8_6V**	**D8**	**D10**
127.6384	127.016	137.1521	240.4248	922.7398

Table 1. Time elapsed (in seconds)

Secondly, Fig. (2) represents graphically the results shown in the table. In this figure, the growth of the dimensions of the problem (shown in the x axis) is not linear, and it may give a false illusion that the growth in time is exponential. Thus, Fig. (3) will be a modification of that figure, in which only the data sets D4, D6, D8 and D10 are shown.

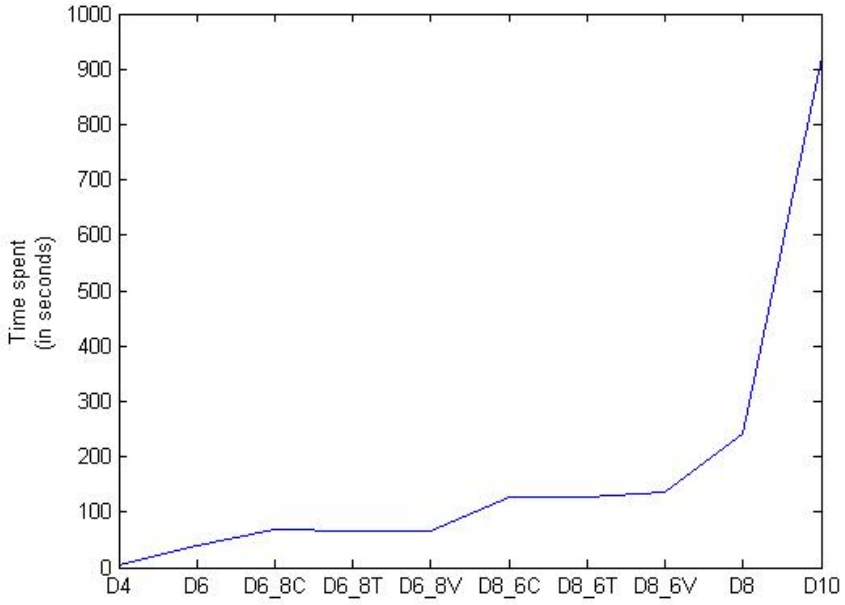

Figure 2. Comparison of time spent finding a solution

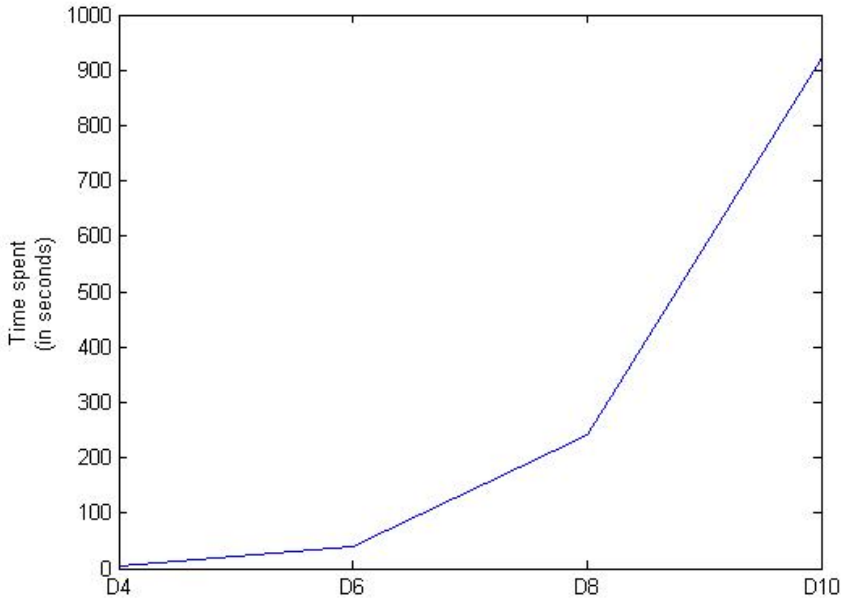

Figure 3. Mean time for finding a solution to D4, D6, D8 and D10

Chapter 5

Discussion

In other investigations, like in [6], there are some graphs and tables showing the time spent for the algorithm with data sets of different sizes. Nonetheless, all the dimensions (except the periods) of the data sets in which the algorithm was run, increased or decreased in the same amount, like in Fig. (3) in this project. Moreover, what is the scalability of the algorithm, is not discussed in them.

Therefore, here it will be discussed if a relation between the size of the problem and the running time of the algorithm can be found. And also, whether increasing or decreasing a specific dimension has more impact on this time than increasing or decreasing a different dimension.

5.1 Analyzing the results

From Fig. (2), it can be clearly appreciated that a change of size in one dimension has the same influence as changing another different dimension (e.g. a schedule for a number of 6 groups, 6 teachers and 8 venues, needs the same time to be found as another for a number of 8 groups, 6 teachers and 6 venues). Therefore, from here it can be extracted that just the total size of the problem ($c \cdot t \cdot v$), ignoring the variation in the number of periods since it is supposed to be constant, is relevant for the time that the algorithm will spend finding a solution.

Respecting the scalability of the problem, it can be seen in Figs. (2) and (3) that the increase of the running time with respect to the size (or dimension) of the problem, is not linear. But it also needs to be considered that, when the size of the problem is increased in all the dimensions (except the periods) in, for example, 2 times, its size is going to increase from $c \cdot t \cdot v \cdot p$ to $2c \cdot 2t \cdot 2v \cdot p$, and therefore, the total increase is $2 \cdot 2 \cdot 2 = 8$, (2^3).

Thus, in order to find a relation, the rate between the computation time of D8 and D4 will be computed, and then since the result of this can be seen as a cubic number (because the three dimensions are being multiplied by 2), the cubic root of this will be also computed:

$$240.4248/4.3340 = 55.4741 \qquad \text{(Op.1)}$$

$$\sqrt[3]{55.4741} = 3.8138 \quad \text{(Op.2)}$$

Now, the same can be done with the running time of D6 and D4. In this case, the three dimensions are being increased by 1.5. Then, the question will be whether there is a relation between the factor 2 and the result of the operations above, and the factor 1.5 and the result of the operations below, or not:

$$39.2513/4.3340 = 9.0566 \quad \text{(Op.3)}$$

$$\sqrt[3]{9.0566} = 2.0844 \quad \text{(Op.4)}$$

So now, having in mind that $2^2 = 4$ and that $1.5^2 = 2.25$, it could be thought, from Ops. (2) and (4), that an increase of a dimension by a factor f would mean that the time spent to get a schedule would be multiplied by a factor around f^2. This hypothesis can be checked with the remaining data sets, by applying the inverse operations. For D6_8*, the factors are 1.5, 1.5, and 2 from D4; or 1, 1, and 1.33 from D6. For D8_6*, they are 1.5, 2, and 2 from D4; or 1, 1.33, and 1.33 from D6. And for D10, they are 2.5, 2.5, and 2.5 from D4; 1.66, 1.66, and 1.66 from D6; or 1.25, 1.25, and 1.25 from D8:

$$D4 * 1.5^2 * 1.5^2 * 2^2 = 87.7635 \quad \text{(Op.5)}$$

$$D6 * 1^2 * 1^2 * 1.33^2 = 69.78 \quad \text{(Op.6)}$$

$$D4 * 1.5^2 * 2^2 * 2^2 = 156.024 \quad \text{(Op.7)}$$

$$D6 * 1^2 * 1.33^2 * 1.33^2 = 124.0535 \quad \text{(Op.8)}$$

$$D4 * 2.5^2 * 2.5^2 * 2.5^2 = 1058.1055 \quad \text{(Op.9)}$$

$$D6 * 1.66^2 * 1.66^2 * 1.66^2 = 841.2916 \quad \text{(Op.10)}$$

$$D8 * 1.25^2 * 1.25^2 * 1.25^2 = 917.1478 \quad \text{(Op.11)}$$

D4, D6, and D8 refer to the mean time over 10 executions shown in Table (1). Operations (5) and (6) estimate the time needed to find a solution to D6_8*; operations (7) and (8) do the same but for the data set D8_6*; and operations (9), (10), and (11) for the data set D10.

Looking at the results of Operations (5) and (6), it can be appreciated that the computing time of D6_8* follows the proposed hypothesis. As well as D8_6* looking at Operations (7) and (8) also has the same tendency. It can be also noticed in Operations (9), (10), and (11), that the real time needed for finding a solution to D10 is around the estimated time obtained from the times needed for D4, D6 and D8, respectively.

Fig. (4) shows a comparison between the mean time that the algorithm took finding a solution for the data sets D6_8*, D8_6*, D8 and D10; and the predicted time using the hypothesis extracted in this chapter, using the data set D4 as the base for the prediction. The double blue line in D6_8* and D8_6* expresses the minimum and maximum mean time in the three variations of these data sets.

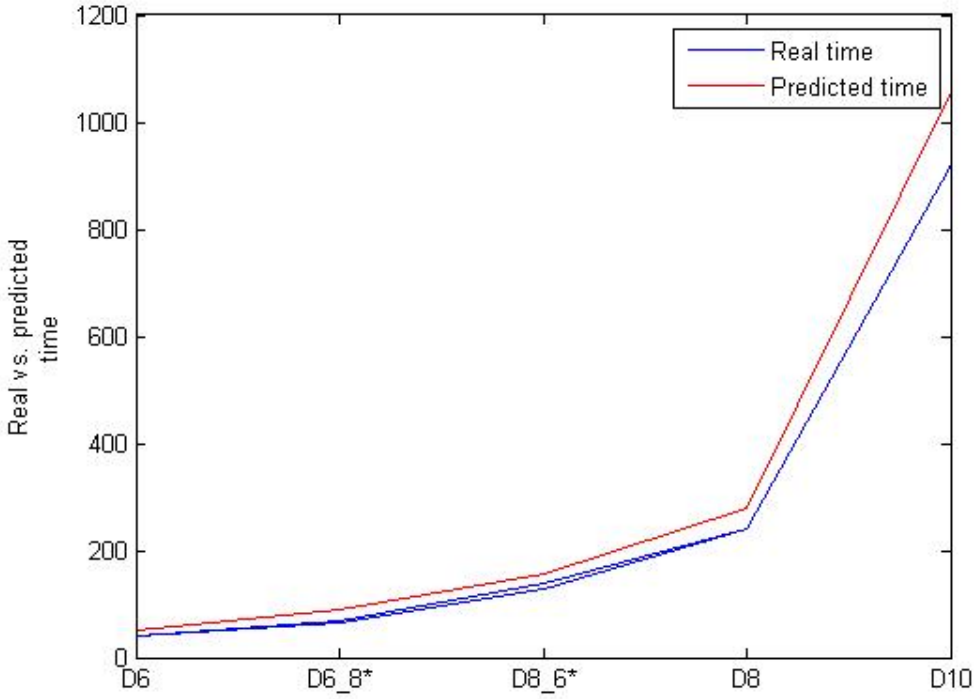

Figure 4. Real time vs. predicted time

Therefore, it looks that this Hopfield Neural Network algorithm could scale quadratically with the size of the problem for every dimension. Nevertheless, there is no statistical evidence for this affirmation, it is just a trend that these limited number of data sets follow. Consequently, the hypothesis cannot be generalized for every data set of any size.

Chapter 6

Conclusions

In reference to the problem statement, at the beginning of this project, a relation between the size of the problem for every dimension and the time needed to run the algorithm has been found, but it is only a hypothesis since there is no statistical evidence that support it. This hypothetical relation can be expressed as a polynomial quadratic function, which means that it could be a good alternative to the traditional approaches to the timetabling problem (which usually is a NP-complete problem, as stated in Section 1) if the hypothesis were confirmed.

It was tried to execute the algorithm on a bigger problem (16 groups, 16 teachers and 16 venues), but it did not fit in memory. The matrix that takes the most space in memory is the matrix of weights W, and since MATLAB uses 8 Bytes of memory for a double value, this matrix would use 112.5 GB in this case. One solution to this that can be tried in future work could be using sparse matrices as proposed in [6].

Also, it could be interesting to apply this method to more realistic problems and evaluate how it behaves. This could be, using the algorithm with real schools requirements, and check whether the algorithm scales as in the hypothesis (considering that the algorithm uses sparse matrices, since probably real data sets are bigger than the example that did not fit), and if it is reasonable to replace manual timetabling by this method. If so, the algorithm could be used to avoid the effort that is needed every year for doing this task [7].

Another interesting topic of investigation could be applying this method to other kinds of scheduling (job shop scheduling, public transport scheduling, broadcast scheduling, etc.). And then study whether its scalability is good for those problems.

Bibliography

[1] H. Aytug, S. Bhattacharyya, G.J. Koehler, and J.L. Snowdown, “A review of Machine Learning in Scheduling”, *IEE Transactions on Engineering Management*, Vol. 41, No. 2, pp. 165-171, May 1994

[2] C.-Y. Lee , S. Piramuthu & Y.-K. Tsai, “Job shop scheduling with a genetic algorithm and machine learning”, *International Journal of Production Research*, Vol. 35, No. 4, pp. 1171-1191, 1997, `http://dx.doi.org/10.1080/002075497195605`

[3] Michael L. Pinedo. *Scheduling. Theory, Algorithms and Systems*, Fourth Edition, pp. 491-496, 2012.

[4] Mark S. Fox, Bradley P. Allen, Stephen Smith, and Gary A. Strohm, “ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling System Summary”, *tech. report CMU-RI-TR-83-08, Robotics Institute, Carnegie Mellon University*, June 1983.

[5] A.H. Karami, M. Hasanzadeh, "University course timetabling using a new hybrid genetic algorithm," *2012 2nd International eConference on Computer and Knowledge Engineering (ICCKE)*, pp.144-149, 18-19 October 2012.

[6] Kate A. Smith, David Abramson, David Duke, “Hopfield neural networks for timetabling: formulations, methods, and comparative results”, *Computers & Industrial Engineering*, Vol. 44, Issue 2, pp. 283-305, February 2003.

[7] Edmund Burke, Dave Elliman, Peter Ford, Rupert Weare, “Examination timetabling in British Universities: A survey”, *Practice and Theory of Automated Timetabling*, pp. 76-90, 1996

[8] M. Doulaty, M. R. Feizi Derakhshi, and M. Abdi, "Timetabling: A State-of-the-Art Evolutionary Approach," *International Journal of Machine Learning and Computing*, Vol. 3, no. 3, pp. 255-258, 2013

[9] M.P. Carrasco, M.V. Pato, “A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem”, *European Journal of Operational Research*, Vol. 153, Issue 1, pp. 65-79, 16 February 2004

[10] Helmut E. Mausser, Michael J. Magazine, "Comparison of neural and heuristic methods for a timetabling problem", *European Journal of Operational Research*, Vol. 93, Issue 2, pp. 271-287, 6 September 1996